TECHNICAL MEMORANDUM (NASA) 77

BI-DIRECTIONAL COMMUNICATION INTERFACE

FOR MICROPROCESSOR-TO-SYSTEM/370

Described is a hardware and software interface to allow two-way communication between a microprocessor system and the IBM System/370

(NASA-CR-163940) BI-DIRECTIONAL COMMUNICATION INTERFACE FOR MICROPROCESSOR-TO-SYSTEM/370 (Ohio Univ.) 24 p HC A02/MF A01 CSCL 09B

N81-17717

Unclas
G3/60 41416

by

Joseph P. Fischer

Avionics Engineering Center
Department of Electrical Engineering
Ohio University
Athens, Ohio 45701

January 1981

Supported by

National Aeronautics and Space Administration
Langley Research Center
Hampton, Virginia
Grant NGR 36-009-017

I. INTRODUCTION

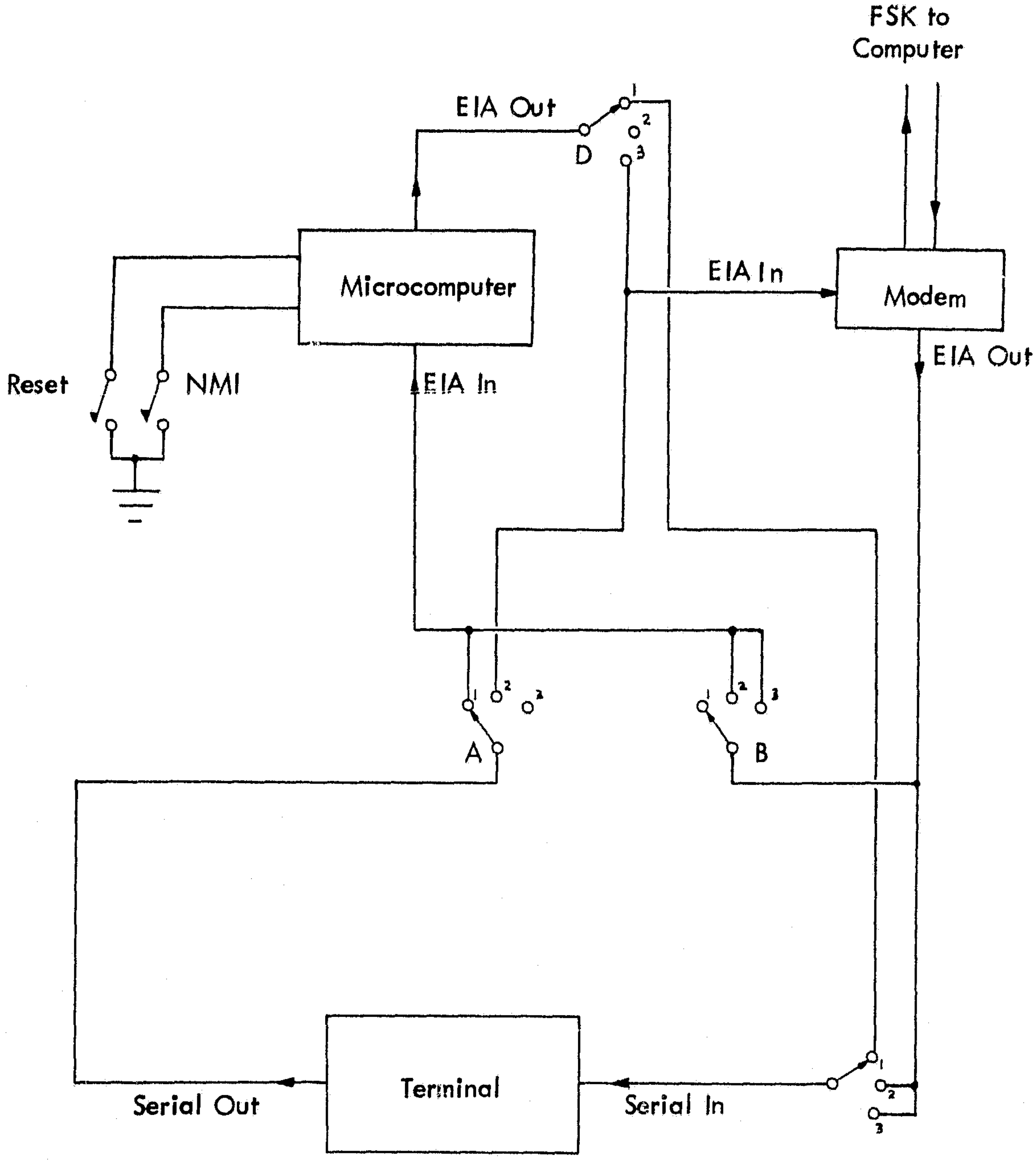
This paper documents the design and operation of a bi-directional communication interface between a microcomputer and the IBM System/370. The hardware unit inter-connects a modem to interface to the S/370, the microcomputer with an EIA I/O port, and a terminal for sending and receiving data from either the microcomputer or the S/370. Also described is the software necessary for the two-way interface. This interface has been designed so that no modifications need to be made to the terminal, modem, or microcomputer. This unit is designed to upgrade an uni-directional interface already in use [1]

II. INTERFACE DESCRIPTION: HARDWARE

Figure 1 shows the paths of signals between the microcomputer, the modem, and the terminal. The hardware interface consists of a four-pole, three-position switch and cables and plugs to connect the switch box to the other devices. All signals are assumed to be RS-232C (EIA standard).

In switch position 1, the microcomputer is connected directly to the terminal; all communications are between these two only. The modem is isolated in this position and it is not necessary to have it connected if no communication to the S/370 is desired. In position 2, the serial out from the keyboard is routed to the modem for communicating to the S/370. The serial out from the modem goes to the terminal and the serial in of the microcomputer. In this position, it is possible to send commands and receive responses from the S/370, while the microcomputer reads the data sent by the S/370. Thus it is possible to load a program into the microcomputer by displaying the object file on terminal. It is necessary to switch to position 1 and issue the microcomputer load command prior to typing the file. Position 3 on the switch box connects the serial out from the modem to the terminal and to the serial in on the microcomputer. In addition, the serial out from the microcomputer is sent to the modem. Here, the microcomputer communicates directly with the S/370, the terminal always displays the response sent by the S/370. With proper positioning of the half-duplex/ full-duplex switches on the terminal and modem, the responses from the microcomputer may also be displayed. Note that the serial-out from the terminal is isolated, thus it may be necessary to start a program on the microcomputer by pressing the NMI (non-maskable interrupt) switch on the switch box.

Table 1 lists the connection used on the terminal and modem. Connections for RS-232C are made through 25-pin D-connectors. Data terminal equipment (DTE) devices are supplied with a male (DB-25P) connector while data communication equipment (DCE) devices are supplied with a female (DB-25S) connector. Figure 2 shows the detailed routing of connections from the connectors on the terminal and modem through the switch box.



Note: Switches Shown in Position 1.

Figure 1. Signal Routing For Bi-Directional Interface.

	DTE		DCE	
P GND	1	—	1	Protective Ground
S OUT	2	→	2	Serial Output
S IN	3	←	3	Serial Input
RTS	4	→	4	Request to Send
CTS	5	←	5	Clear To Send
DSR	6	←	6	Data Set Ready
S GND	7	—	7	Signal Ground
RLSD	8	←	8	Received Line Signal Detect
DTR	20	→	20	Data Terminal Ready

Table 1. RS-232C Connections.

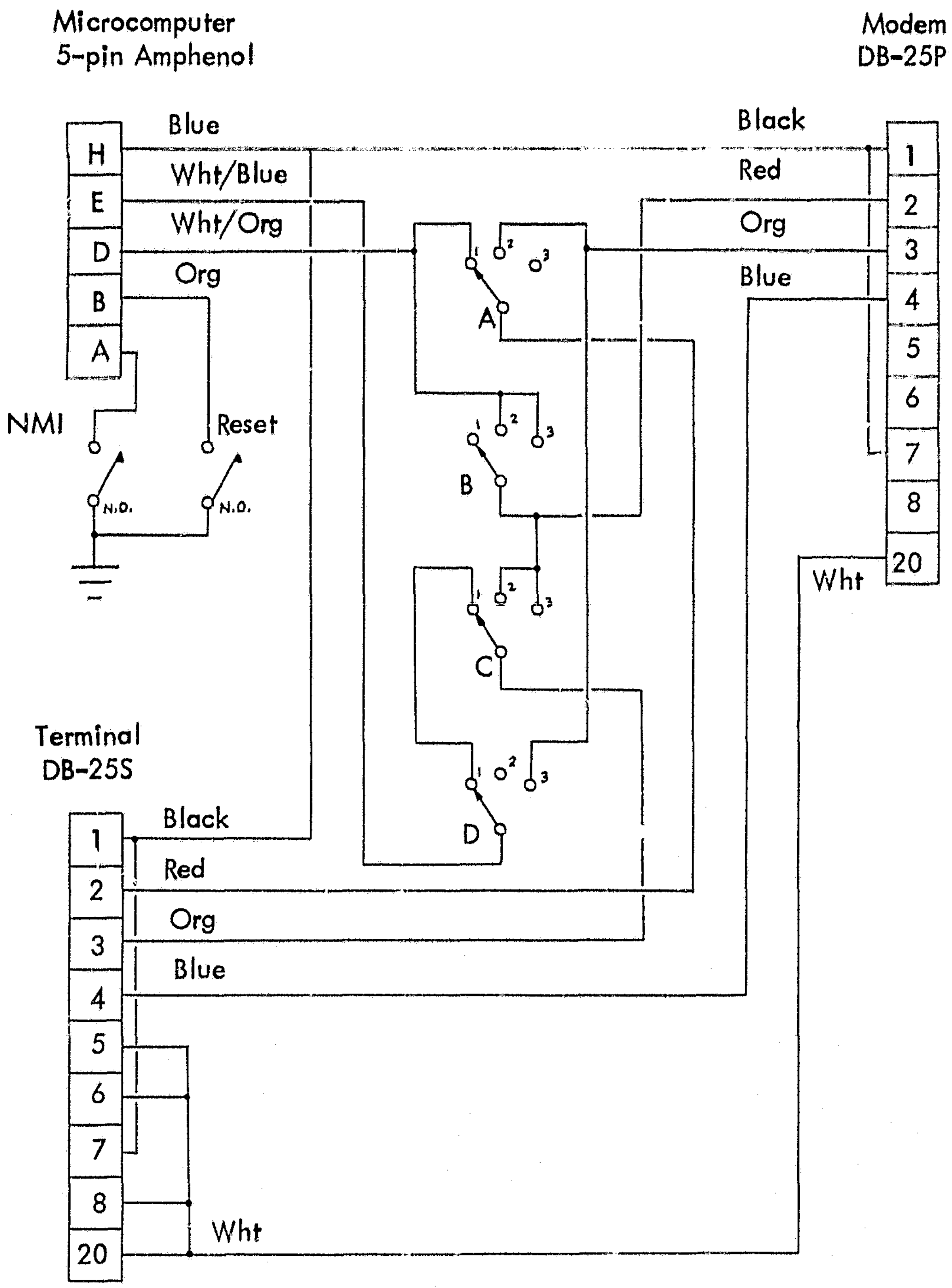


Figure 2. Interface Wiring Diagram.

III. INTERFACE DESCRIPTION: SOFTWARE

Full utilization of the bi-directional interface requires a set of programs to be run simultaneously on the microcomputer and the S/370. Figure 3 shows a block diagram of how the programs would operate for a typical application. Some points to be considered in writing the interface software are:

a. Most microcomputers store character data internally as ASCII.

b. Serial communications between devices are generally in ASCII format.

c. The I/O routines for the S/370 expect to receive ASCII which is then converted to EBCDIC, which the S/370 uses for internal storage of character data.

d. The Conversational Monitor System (CMS) portion of the VM/370 operating system is line-oriented, i.e., no system action is taken until a carriage return (hex OD) is received.

e. The S/370 issues a prompt when ready for another line.

A typical application for which this interface has been used is transmitting data collected by the microcomputer on a cassette tape to the S/370, where it is stored on a disk file for further processing. The sequence of events is as follows: the data to be transmitted is stored in a buffer in the microcomputer's memory. Generally, 80 characters comprise one line. Note that one byte consists of two four-bit hexadecimal numbers, each of which is converted to ASCII. Thus if 80 characters are to be sent, the buffer is 40 bytes long. After 80 characters are sent, a carriage return (hex OD) is sent. The S/370 does the ASCII-to-EBCDIC conversion and places the EBCDIC characters in a user buffer in the S/370 memory. When the S/370 is ready to receive another line, it sends a series of control characters. The microcomputer reads and recognizes these control characters as the prompt signal to send another line. The sequence of control characters currently sent by S/370 is shown in Figure 4.

Appendix A gives a listing of a MOS Technology 6502 microcomputer program (intended to be run using the 'Super-Jolt' micro unit) for reading 40 bytes of data from a Memodyne digital cassette tape unit and sending these to the S/370. The data to be sent are packed BCD numbers; i.e., one BCD digit occupies four bits, two BCD numbers are contained in one byte. Each BCD digit is sent as ASCII by the 'output byte' routine in the microcomputer monitor program (at address 72B1 (hex) in the Super-Jolt (TM) monitor). A carriage return is sent at the end of the line with a call to the WRT routine at address 72C6 (hex).

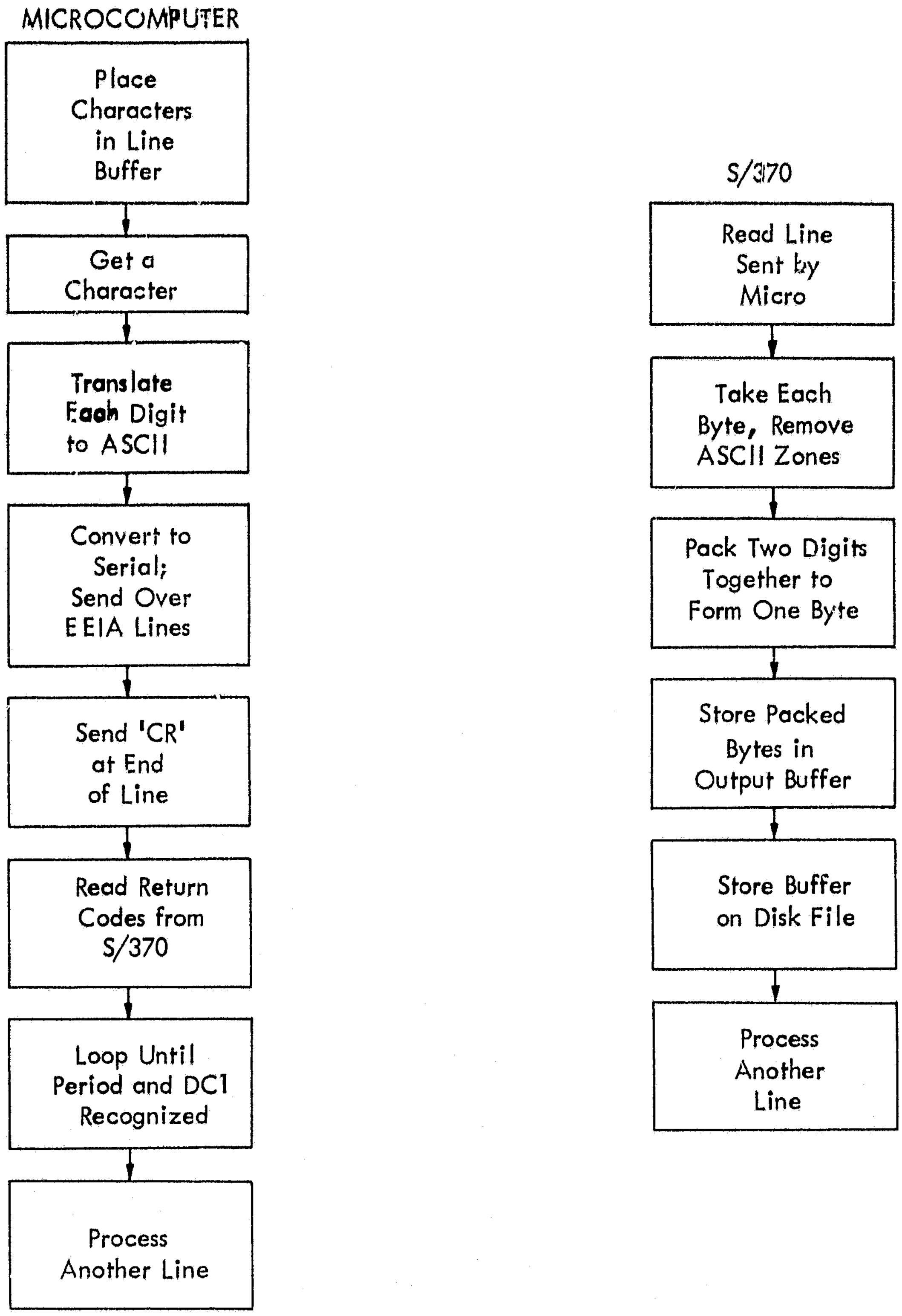


Figure 3. Control Program Flow Charts.

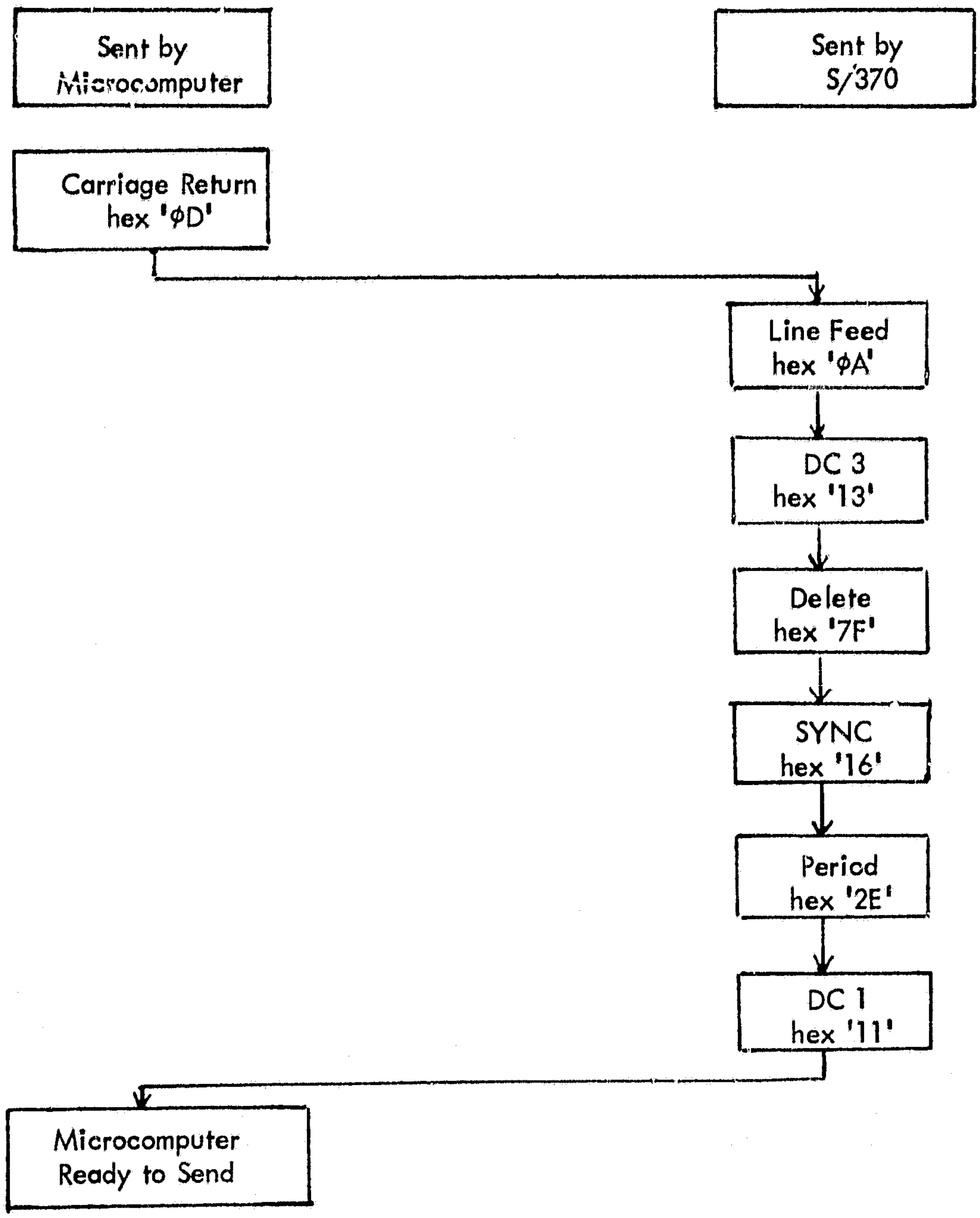


Figure 4. Control Characters Sent by S/370 After Receiving Carriage Return.

Program lines 1 to 78 are initialization steps used for the Memodyne interface hardware and to position the tape properly. Lines 79 to 91 constitute the main part of the program which builds up the 40-byte buffer then sends the buffer to the S/370. This part loops continuously whether or not any data is received. The operator should monitor the operation to stop the program when all the data has been transmitted. Subroutine READ is called to read a byte from the tape unit. Subroutine W370 sends the 40-character buffer to the S/370, sends a carriage return, then looks for a period (hex 2E) followed by a DC-1 (hex 11). If this sequence is not done, the S/370 issues a read-error message. When these two characters are received, control is passed back to the main program sequence.

Subroutine RDT is a modification of the RDT routine at address 72E9 in the Jolt monitor. Most serial-read routines on microcomputers are full-duplex; as each bit is received, it is echoed back out to the sending device. However, the S/370 can receive half-duplex only. Thus it is necessary to change the interface method through the modem or to re-write the read routine so that the received bits are not echoed by the microcomputer. This is the purpose of having a separate read routine. If this is not done, read-errors result. The program presented here is shown to illustrate one application of the bi-directional interface. Other uses on other microcomputers would still use the same basic philosophy.

The companion program that is run on the S/370 is shown in Appendix B. This program is written in IBM 360/370 assembler language [2] using standard CMS I/O routines. Again this program illustrates the application of sending data to the S/370 for storage on a disk file.

The data is read 80-bytes at a time, each BCD character in its ASCII format. Each character read is stripped of the upper four-digit mask and is repacked. This is done by the translate instruction at line 88 and the PACK instruction at line 90. Since one record produces only 40 packed BCD digits, two lines are read before one 80-byte record is written to the file. A blank line or an incomplete line is filled to the end with zeros. Each time a record is written, a counter is incremented which is printed at the end of program execution.

IV. INTERFACE OPERATION

The example of transmitting data from the microcomputer to the S/370 will be continued here to show how the interface may be operated. After the interface is properly connected, power should be applied to all units. At this point it is usually necessary to load the microcomputer with a program stored on a disk file. Thus the switch box should be set to position 2 and the appropriate CMS LOGON procedure performed. When the microprocessor object code is ready for transmittal (through editting, assembling, simulating, etc.) the switch box should be set to position 1, the microcomputer reset button pushed, and a carriage return or other appropriate

key to reset the microcomputer typed. Then issue the proper command to set the microcomputer for loading hexadecimal data over its serial lines. The switch box is then set back to position 2 and the appropriate command is issued to the S/370 to load the microcomputer with the object file. Next, the unit is switched back to position 1 to verify correct loading, initialize any memory locations and set up the NMI vector address to the start of the program. Now the switch box is placed in position 2 and the program to receive the data is started and then the unit is set to position 3 and the NMI button pressed.

As operation commences, the prompting period and any other responses from the S/370 will be displayed on the terminal. Depending on the setting of the half-duplex/full-duplex switches on the terminal and modem, data sent by the microcomputer will also be displayed on the terminal.

When the operation is finished, the unit may be set to position 2 to stop the S/370 program then position 1 to stop the microcomputer program.

V. SUMMARY

A discussion was presented here of an interface unit and software procedures to allow two-way communication between a microcomputer and a central computer. This can be used for two-way data transmission, control and other applications where bi-directional communications are necessary. As an aid to setting up the software for other computer systems, ASCII [3] and EBCDIC [4] tables are given in Tables 2 and 3.

	000	001	010	011	100	101	110	111
0000	NULL	① DC_0	b̸	0	@	P		
0001	SOM	DC_1	!	1	A	Q		
0010	EOA	DC_2	"	2	B	R		
0011	EOM	DC_3	#	3	C	S		
0100	EOT	DC_4 (stop)	$	4	D	T		
0101	WRU	ERR	%	5	E	U		
0110	RU	SYNC	&	6	F	V		
0111	BELL	LEM	'	7	G	W	Unassigned	
1000	FE_0	S_0	(	8	H	X		
1001	HT / SK	S_1	)	9	I	Y		
1010	LF	S_2	*	:	J	Z		
1011	V_{TAB}	S_3	+	;	K	[		
1100	FF	S_4	(comma) ,	<	L	\		ACK
1101	CR	S_5	−	=	M	]		②
1110	SO	S_6	.	>	N	↑		ESC
1111	SI	S_7	/	?	O	←		DEL

Example: | 100 | 0001 | = A

b_7 ———————— b_1

The abbreviations used in the figure mean:

NULL	Null Idle	CR	Carriage return
SOM	Start of message	SO	Shift out
EOA	End of address	SI	Shift in
EOM	End of message	DC_0	Device control ① Reserved for data Link escape
EOT	End of transmission	DC_1 - DC_3	Device control
WRU	"Who are you?"	ERR	Error
RU	"Are you . . . . ?"	SYNC	Synchronous idle
BELL	Audible signal	LEM	Logical end of media
FE	Format effector	SO_0 - SO_7	Separator (information)
HT	Horizontal tabulation		Word separator (blank, normally non-printing)
SK	Skip (punched card)	ACK	Acknowledge
LF	Line feed	②	Unassigned control
V/TAB	Vertical tabulation	ESC	Escape
FF	Form feed	DEL	Delete Idle

Table 2. ASCII Table.

Bit Positions 4, 5, 6, 7	Second Hexadecimal Digit	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Bit Positions 0,1		00				01				10				11			
Bit Positions 2,3		00	01	10	11	00	01	10	11	00	01	10	11	00	01	10	11
0000	0	NUL (1)	DLE (2)	DS (3)	(4)	SP (5)	& (6)	- (7)	(8)					{ (9)	} (10)	\ (11)	0 (12)
0001	1	SOH	DC1	SOS				/ (13)		a	j	~		A	J	(14)	1
0010	2	STX	DC2	FS	SYN					b	k	s		B	K	S	2
0011	3	ETX	TM							c	l	t		C	L	T	3
0100	4	PF	RES	BYP	PN					d	m	u		D	M	U	4
0101	5	HT	NL	LF	RS					e	n	v		E	N	V	5
0110	6	LC	BS	ETB	UC					f	o	w		F	O	W	6
0111	7	DEL	IL	ESC	EOT					g	p	x		G	P	X	7
1000	8	GE	CAN							h	q	y		H	Q	Y	8
1001	9	RLF	EM						`	i	r	z		I	R	Z	9
1010	A	SMM	CC	SM		¢	!	¦ (15)	:								|
1011	B	VT	CU1	CU2	CU3	.	$	,	#								
1100	C	FF	IFS		DC4	<	*	%	@					⑀		⑁	
1101	D	CR	IGS	ENQ	NAK	(	)	_	'								
1110	E	SO	IRS	ACK		+	;	>	=					⑂			
1111	F	SI	IUS	BEL	SUB	|	¬	?	"								EO

First Hexadecimal Digit; Zone Punches; Digit Punches

Card Hole Pattern

(1) 12-0-9-8-1
(2) 12-11-9-8-1
(3) 11-0-9-8-1
(4) 12-11-0-9-8-1
(5) No Punches
(6) 12
(7) 11
(8) 12-11-0
(9) 12-0
(10) 11-0
(11) 0-8-2
(12) 0
(13) 0-1
(14) 11-0-9-1
(15) 12-11

Control Character Representations

ACK	Acknowledge
BEL	Bell
BS	Backspace
BYP	Bypass
CAN	Cancel
CC	Cursor Control
CR	Carriage Return
CU1	Customer Use 1
CU2	Customer Use 2
CU3	Customer Use 3
DC1	Device Control 1
DC2	Device Control 2
DC4	Device Control 4
DEL	Delete
DLE	Data Link Escape
DS	Digit Select
EM	End of Medium
ENQ	Enquiry
EO	Eight Ones
EOT	End of Transmission
ESC	Escape
ETB	End of Transmission Block
ETX	End of Text
FF	Form Feed
FS	Field Separator
GE	Graphic Escape
HT	Horizontal Tab
IFS	Interchange File Separator
IGS	Interchange Group Separator
IL	Idle
IRS	Interchange Record Separator
IUS	Interchange Unit Separator
LC	Lower Case
LF	Line Feed
NAK	Negative Acknowledge
NL	New Line
NUL	Null
PF	Punch Off
PN	Punch On
RES	Restore
RLF	Reverse Line Feed
RS	Reader Stop
SI	Shift In
SM	Set Mode
SMM	Start of Manual Message
SO	Shift Out
SOH	Start of Heading
SOS	Start of Significance
SP	Space
STX	Start of Text
SUB	Substitute
SYN	Synchronous Idle
TM	Tape Mark
UC	Upper Case
VT	Vertical Tab

Special Graphic Characters

¢	Cent Sign
.	Period, Decimal Point
<	Less-than Sign
(	Left Parenthesis
+	Plus Sign
|	Logical OR
&	Ampersand
!	Exclamation Point
$	Dollar Sign
*	Asterisk
)	Right Parenthesis
;	Semicolon
¬	Logical NOT
-	Minus Sign, Hyphen
/	Slash
¦	Vertical Line
,	Comma
%	Percent
_	Underscore
>	Greater-than Sign
?	Question Mark
`	Grave Accent
:	Colon
#	Number Sign
@	At Sign
'	Prime, Apostrophe
=	Equal Sign
"	Quotation Mark
~	Tilde
{	Opening Brace
⑀	Hook
⑂	Fork
}	Closing Brace
\	Reverse Slant
⑁	Chair
|	Long Vertical Mark

Table 3. EBCDIC Table.

VI. PEFERENCES

[1] Microprocessor-to-System/370 Interface, Robert W. Lilley, NASA TM-55 (Revised), Avionics Engineering Center, Department of Electrical Engineering, Ohio University, April, 1978.

[2] OS/VS - DOS/VS - VM/370 Assembler Language, GC33-4010-4, International Business Machines Corporation, February, 1975.

[3] Deem, Bill R., Kenneth Muchow, and Anthony Zeppa, "Digital Computer Circuits and Concepts", Reston Publishing Company, Inc., Reston Virginia, 1974, pg. 56.

[4] IBM System/370 Principles of Operation, GA22-7000-5, International Business Machines Corporation, August 1976, pg. 288.

VII. APPENDICES

A. Program Listing for Microcomputer Control Program.

```
**********************************************************************
*                                                                    *
*      THIS PROGRAM IS DESIGNED FOR RUNNING ON THE JOLT/MEMODYNE     *
*      SYSTEM FOR RECOVERING DATA STORED ON THE DIGITAL TAPE.        *
*      THE DATA IS READ IN 40 BYTES AT A TIME AND STORED IN A        *
*      BUFFER, THEN THE BUFFER IS SENT TO THE S/370 OVER THE JOLT'S  *
*      SERIAL LINES.  WITH ASCII CONVERSION, 80 BYTES ARE ACTUALLY   *
*      SENT OVER THE SERIAL LINES.                                   *
*                                                                    *
*          J.P. FISCHER        08/1980                               *
*                                                                    *
**********************************************************************
*
*
PIAA   EQU $4000    ADDRESS OF PIA SIDE A
PIAB   EQU $4002    ADDRESS OF PIA SIDE B
WRT    EQU $72C6    JOLT WRITE DATA TO SERIAL OUT LINE
WROB   EQU $72B1
MPB    EQU $6E02    PIA B FOR SERIAL I/O WORK
MCLK1T EQU $6E04    PIA TIMER
MCLKRD EQU $6E04    SAME AS ABOVE
MCLK1F EQU $6E05    SOME MORE TIMER STUFF
MAJCRT EQU $EA      UPPER 8 BITS OF BAUD RATE
MINCRT EQU $EB      LOWER 8 BITS OF BAUD RATE
TAPESY EQU %00000010  PATTERN FOR TAPE SYNC CHECK
BOT    EQU %00000100  PATTERN FOR BOT/EOT CHECK
LF     EQU %00010000  PATTERN FOR LOAD FOWARD FUNCTION
REW2   EQU %00100000  PATTERN FOR REWIND OPERATION
START  EQU %10000000  PATTERN FOR INITIATING START
*
       ORG 0
XTEMP  BSS 1        TEMPORARY FOR X
YTEMP  BSS 1        TEMPORARY FOR Y
BUFFER BSS 40
*
*
       ORG $200
       JSR INIT     SET UP PIA FOR MEMODYNE
       LDA PIAB     PREPARE TO CHECK BOT
       AND =BOT     SEE IF ON LEADER
       BNE NOTBOT   IF NOT, THEN OK
       LDA PIAB     GET SIDE B
       EOR =LF      CLEAR LOAD FOWARD BIT
       ORA =START   SET START BIT HIGH
       STA PIAB     AND STORE TO LOAD FOWARD
BTLOOP LDA PIAB     GET STATUS
       AND =BOT     SEE IF STILL ON LEADER
       BEQ BTLOOP   CONTINUE TESTING UNTIL OFF
       LDA PIAB
       ORA =LF      SET LOAD FOWARD HIGH
       STA PIAB     AND REPLACE
LDLOOP LDA PIAB
       AND =BOT     NOW LOOP UNTIL AT READY POINT
       BNE LDLOOP   KEEP GOING UNTIL ON HOLE
       LDA PIAB
```

```
        EOR =LF     SET LOAD FOWARD IOW TO MOVE
        STA PIAB    OFF OF HOLE
TLOOP   LDA PIAB
        AND =BOT
        BEQ TLOOP
        LDX =$80    TIMER ROUTINE
TIMXT   DEX
        BNE TIMXT   KEEP LOOPING UNTIL OUT
        ORA =LF     NOW RETURN LOAD FOWARD
        STA PIAB    HIGH, SHOULD BE OFF OF HOLE
NOTBOT  LDA =0      CLEAR ACCUM. AND SET
        STA PIAA+1
        STA PIAB+1
        LDA =$F8
        STA PIAB
        LDA =0
        STA PIAA
        LDA =$FF
        STA PIAB+1
        STA PIAA+1
        LDA PIAB
        AND =%11110111
        STA PIAB
*
*           NOW INITIALIZE THE 370 AND START
*           SENDING DATA.
*
LFLDS   LDX =0      READ 40 CHARACTERS FROM TAPE
L80     JSR READ    GET A BYTE FROM RECORDER
        STA BUFFER,X  SAVE IN OUTPUT BUFFER
        INX         DO ANOTHER ONE
        CPX =40     DONE 40 BYTES YET?
        BNE L80     IF NOT, DO AGAIN
        LDY =40     SEND THESE 40
        JSR W370    SEND TO SYSTEM
        JMP LFLDS
*
*           INITIALIZATION FOR PIA
*
INIT    LDX =0
        STX PIAA+1
        STX PIAB+1
        STX PIAA
        LDA =$B8
        STA PIAB
        LDA =$FF
        STA PIAA+1
        STA PIAB+1
        LDA =0
        ORA =LF
        ORA =REW2
        STA PIAB
        RTS
*
*
```

```
***********************************************************************
*                                                                     *
*      THIS IS THE READING PORTION OF THE PROGRAM TO RECOVER          *
*      DATA FROM THE RECORDER AND PLACE IN THE MICROCOMPUTER'S        *
*      MEMORY.                                                        *
*                                                                     *
***********************************************************************
*
*
READ    LDA PIAB
        ORA =START   SET START HIGH
        STA PIAB
RDLP    LDA PIAB
        AND =TAPESY WAIT UNTIL SYNC IS HIGH
        BEQ RDLP
        LDA PIAB
        EOR =START   SET START LOW AGAIN
        STA PIAB
INLP1   LDA PIAB
        AND =TAPESY WAIT UNTIL SYNC IS LOW
        BNE INLP1
        LDA PIAA     GET THE DATA FROM RECORDER
        RTS
*
*
***********************************************************************
*                                                                     *
*      THIS SUBROUTINE OUTPUTS A LINE OF CHARACTERS TO THE S/370.     *
*      THE ADDRESS OF THE BUFFER IS IN PAGE ZERO AND IS               *
*      INDEXED BY THE X-REGESTER.  THE LENGTH OF THE BUFFER           *
*      TO BE SENT IS CONTAINED IN THE Y-REGESTER.  AFTER THE          *
*      BUFFER IS SENT, A 'CR' IS SENT THEN THE PROGRAM WAITS          *
*      FOR THE CONTROL CHARACTERS BETWEEN THE 'CR' AND PERIOD         *
*      TO BE SENT BACK, THEN WAITS FOR THE CONTROL                    *
*      CHARACTER AFTER THE PERIOD INDICATING THE S/370                *
*      IS IN THE READ STATE.                                          *
*                                                                     *
***********************************************************************
*
W370    LDX =0       POINT TO FIRST CHARACTER
        STX XTEMP    ZERO X-TEMP SPACE
        STY YTEMP    SAVE LENGTH
WLOOP   LDX XTEMP    GET POINTER
        LDA BUFFER,X  GET A CHARACTER
        JSR WROB     SEND IT
        INC XTEMP    X+1
        DEC YTEMP    LESS ONE CHARACTER
        BNE WLOOP    GO AGAIN IF NOT DONE
        LDA =$D      CARRIAGE RETURN
        JSR WRT      TELL 370 THIS IS END-OF-LINE
SCANP   JSR RDT      READ JUNK FROM SYSTEM
        CMP =$2E     PERIOD
        BNE SCANP
        JSR RDT      LOOK FOR
        CMP =$11     DC1
```

```
       BNE SCANP
       RTS          RETURN TO CALLER
*
*
*********************************************************************
*                                                                   *
*         HIGH SPEED REWIND.                                        *
*                                                                   *
*********************************************************************
*
       JSR INIT
       LDA =$B8
       STA PIAB
       LDA =LF
       STA PIAB
       ORA =REW2
       STA PIAB
       BRK
*
*
*********************************************************************
*                                                                   *
*      MODIFIED JOLT READ ROUTINE.                                  *
*      THIS ROUTINE IS IDENTICAL TO THE ORIGINAL 'RDT' ROUTINE      *
*      AT ADDRESS $72E9, BUT THIS ROUTINE OPERATES IN HALF-         *
*      DUPLEX RATHER THAN FULL-DUPLEX MODE.                         *
*                                                                   *
*********************************************************************
*
RDT    LDX =8
*
RDT1   LDA MPB      WAIT FOR START BIT
       LSR A
       BCC RDT1
*
       JSR DLY1
*
RDT2   JSR DLY2
       LDA MPB      CY = NEXT BIT
       LSR A
*
       PHP          SAVE BIT
       TYA          Y CONTAINS CHAR BEING FORMED
       LSR A
       PLP          RECALL BIT
       BCC RDT4
       ORA =$80     ADD IN NEXT BIT
RDT4   TAY
       DEX
       BNE RDT2     LOOP FOR 8 BITS
       EOR =$FF     COMPLEMENT DATA
       AND =$7F     CLEAR PARITY
       JSR DLY2
       CLC
*
```

```
DLY2    JSR DLY1
*
DLY1    PHA         SAVE FLAGS AND A
        PHP
        TXA
        PHA         SAVE X
        LDX MAJCRT
        LDA MINCRT
*
        STA MCLK1T
DL3     LDA MCLK1F
        BPL DL3
        DEX
        PHP
        LDA MCLKRD  RESET TIMER INT FLAG
        PLP
        BPL DL3
*
        PLA
        TAX
        PLP
        PLA
        RTS
*
*
        ORG $FFFA
        HEX 00,02
*
        END
```

B. Program Listing for S/370 Control Program.

```
         TITLE 'UNLOAD$S:  READS RECORDS FROM MEMODYNE/MICROCOMPUTER IN*
               TERFACE AND STORES ON DISK.'
         PRINT NOGEN
         SPACE
***********************************************************************
*                                                                     *
*        THIS PROGRAM IS DESIGNED TO BE RUN ON THE S/370 IN CON-     *
*        JUCTION WITH THE MICRO 'UNLOAD' PROGRAM AND THE MEMODYNE/    *
*        MICROCOMPUTER HARDWARE INTERFACE.  RECORDS READ FROM         *
*        TAPE BE THE MICRO ARE SENT TO THE 370 IN ASCII, CP THEN      *
*        TRANSLATES THESE TO EBCDIC WHICH MUST BE TRANSLATED          *
*        BACK TO HEX BY THIS PROGRAM.  80 BYTES ARE SENT AT A         *
*        TIME (40 EQUIVALENT HEX CHARACTERS) AND 80 HEX CHARACTERS    *
*        ARE STORED ON THE DISK FILE.                                 *
*                                                                     *
*              J. P. FISCHER      08/1980                             *
*                                                                     *
***********************************************************************
         SPACE 2
UNLOAD$S START X'E000'
         USING UNLOAD$S,12
         MVI   FLAGS,0          CLEAR ALL FLAG BITS
         LA    1,8(,1)          POINT TO FILE NAME FIELD
         LR    2,1              SAVE PLIST ADDRESS
         CLI   0(1),X'FF'       BLANK ?
         BE    NOID             IF SO, ERROR
         LA    1,8(,1)
         CLI   0(1),X'FF'       NO FILETYPE?
         BE    NOID             IF NOT, ERROR
         MVC   FILEID+8(16),0(2)    MOVE PARTIAL ID
         LA    1,8(,1)
         CLI   0(1),X'FF'       NO FILEMODE
         BE    NOMODE           IF NOT SUBSTITUTE 'A'
         MVC   FILEID+24(2),16(2)  MOVE IN NEW MODE
         B     CHECK            CONTINUE
NOMODE   MVI   FILEID+24,C'A'   MOVE IN 'A'
         MVI   FILEID+25,C' '
         SPACE
CHECK    LA    1,8(,1)          MOVE POINTER UP SOME MORE
         CLI   0(1),X'FF'       SEE IF ANYTHING THERE
         BE    CHECK1           IF NOT, CONTINUE
         CLI   0(1),C'('        SEE IF OPTION
         BNE   PARMERR          IF NOT, BAD PARM
         LA    1,8(,1)          NEXT FIELD
         CLI   0(1),X'FF'       SEE IF BLANK
         BE    CHECK1
         CLC   0(8,1),OPTREP    SEE IF REPLACE OPTION
         BNE   BADOPT           IF NOT, CONTINUE
         OI    FLAGS,1          SET REPLACE BIT
         SPACE
CHECK1   TM    FLAGS,1          SEE IF REPLACE IN EFFECT
         BZ    OPENF            IF NOT,GO ON
         FSERASE FSCB=FILEID
OPENF    FSOPEN FSCB=FILEID     OPEN FOR WRITING
         CL    15,F36           SEE IF INVALID DISK
```

```
         BE    INVDISK
         SPACE 2
***********************************************************************
*                                                                     *
*              THIS PART OF THE PROGRAM CAUSES A TERMINAL             *
*              READ TO GET THE ASCII CHARACTERS, THEN TRANSLATES      *
*              THEM TO HEX AND STORES ON DISK.                        *
*                                                                     *
***********************************************************************
         SPACE
         SLR   4,4
         SLR   7,7             CLEAR RECORD COUNTER
         LA    2,WBUF          WRITE BUFFER ADDRESS
         LA    4,8             LOOP INCREMENT
         L     5,IBUF80        END OF LOOP
RDLOOP   LA    3,IBUF          READ BUFFER ADDRESS
         RDTERM IBUF           GET A RECORD
         LTR   0,0             SEE IF NULL LINE
         BZ    DONE            GO IF IT IS
         WAITT                 WAIT FOR I/O
         TR    IBUF(80),TRTBL  CHANGE TO HEX
STRIPZ   MVC   TEMP(8),0(3)    GET 8 ZONED BYTES
         PACK  TEMP1(5),TEMP(9)     REMOVE THE ZONES
         MVC   0(4,2),TEMP1    PUT PACKED CHARS. IN OUT BUFFER
         LA    2,4(2)          NEXT POSITION IN OUTPUT BUFFER
         BXLE  3,4,STRIPZ      CONTINUE UNTIL WHOLE RECORD DONE
         SPACE
         RDTERM IBUF           GET ANOTHER 80 CHARS.
         LTR   0,0             SEE IF NULL LINE
         BZ    DONE1           GO IF IT IS
         WAITT                 WAIT FOR I/O
         LA    3,IBUF          RE-INITIALIZE POINTER
         TR    IBUF(80),TRTBL
Z1       MVC   TEMP(8),0(3)    GET 8 BYTES
         PACK  TEMP1(5),TEMP(9)     REMOVE ZONES
         MVC   0(4,2),TEMP1    PUT IN OUT BUFFER
         LA    2,4(2)          NEXT LOCATION
         BXLE  3,4,Z1          DO 80 BYTES
         LA    2,WBUF          REINTIALIZE WRITE POINTER
         FSWRITE FSCB=FILEID   SEND TO DISK
         LTR   15,15           SEE IF ERROR
         BNZ   WRTERR          GO IF THERE IS
         LA    7,1(7)          ADD ONE TO RECORD COUNT
         B     RDLOOP          PROCESS SOME MORE
         SPACE
DONE1    MVI   WBUF+40,0       PREPARE TO CLEAR
         MVC   WBUF+41(39),WBUF+40 REMAINING FIELD
         FSWRITE FSCB=FILEID
         LTR   15,15
         BNZ   WRTERR
         LA    7,1(7)          ADD ONE TO RECORD COUNT
         SPACE
***********************************************************************
*                                                                     *
*              NOW CLOSE THE FILE.                                    *
```

```
*                                                                      *
************************************************************************
         SPACE
DONE     FSCLOSE FSCB=FILEID  CLOSE THE FILE
         LINEDIT TEXT='..... RECORDS WRITTEN TO FILE.',                *
               SUB=(DEC,(7)),DOT=NO,RENT=NO
         SLR   15,15           CLEAR RETURN CODE
         BR    14              GO TO CMS
         EJECT
NOID     LINEDIT TEXT='DMSULD054E INCOMPLETE FILEID SPECIFIED.',       *
               DISP=ERRMSG,DOT=NO,RENT=NO
         LA    15,24           RETURN CODE
         BR    14              BACK TO CMS
         SPACE
INVDISK  LR    2,15            SAVE RETURN CODE
         LINEDIT TEXT='DMSULD069E DISK ''..'' NOT ACCESSED.',          *
               SUB=(CHARA,FILEID+24),DISP=ERRMSG,DOT=NO,RENT=NO
         LR    15,2            GET RETURN CODE
         BR    14              BACK TO CMS
         SPACE
WRTERR   LR    2,15            SAVE RETURN CODE
         LINEDIT TEXT='DMSULD105S ERROR ''..'' WRITING FILE ''........ *
               ........ ..'' ON DISK.',SUB=(DEC,(2),CHARA,FILEID+8,CHAR*
               A,FILEID+16,CHARA,FILEID+24),DISP=ERRMSG,               *
               DOT=NO,RENT=NO
         LA    15,100          RETURN CODE
         BR    14              BACK TO CMS
         SPACE
PARMERR  LR    2,1             SAVE PARM ADDRESS
         LINEDIT TEXT='DMSULD070E INVALID PARAMETER ''........''',     *
               SUB=(CHARA,(2)),DISP=ERRMSG,DOT=NO,RENT=NO
         LA    15,24           RETURN CODE
         BR    14              BACK TO CMS
         SPACE
BADOPT   LR    2,1             SAVE OPTION ADDRESS
         LINEDIT TEXT='DMSULD003E INVALID OPTION ''........''',        *
               SUB=(CHARA,(2)),DISP=ERRMSG,DOT=NO,RENT=NO
         LA    15,24           RETURN CODE
         BR    14              BACK TO CMS
         EJECT
         DS    0D
F36      DC    F'36'
IBUF80   DC    AL4(IBUF+79)
OPTREP   DC    CL8'REP'
FILEID   FSCB '* * *',BUFFER=WBUF,BSIZE=80
TEMP     DS    XL8
         DC    C'1'
TEMP1    DS    XL5
FLAGS    DS    XL1
WBUF     DS    XL80
IBUF     DS    XL132
TBL      DC    X'00'
         ORG   TBL+3*16
         DC    XL16'00FAFBFCFDFEFFF0F0F0F0F0F0F0F0F0'
         ORG   TBL+11*16
```

```
         DC    XL16'F0F1F2F3F4F5F6F7F8F9F0F0F0F0F0F0'                  UNL01660
TRTBL    EQU   TBL-X'40'                                               UNL01670
         END   UNLOAD$S                                                UNL01680
```